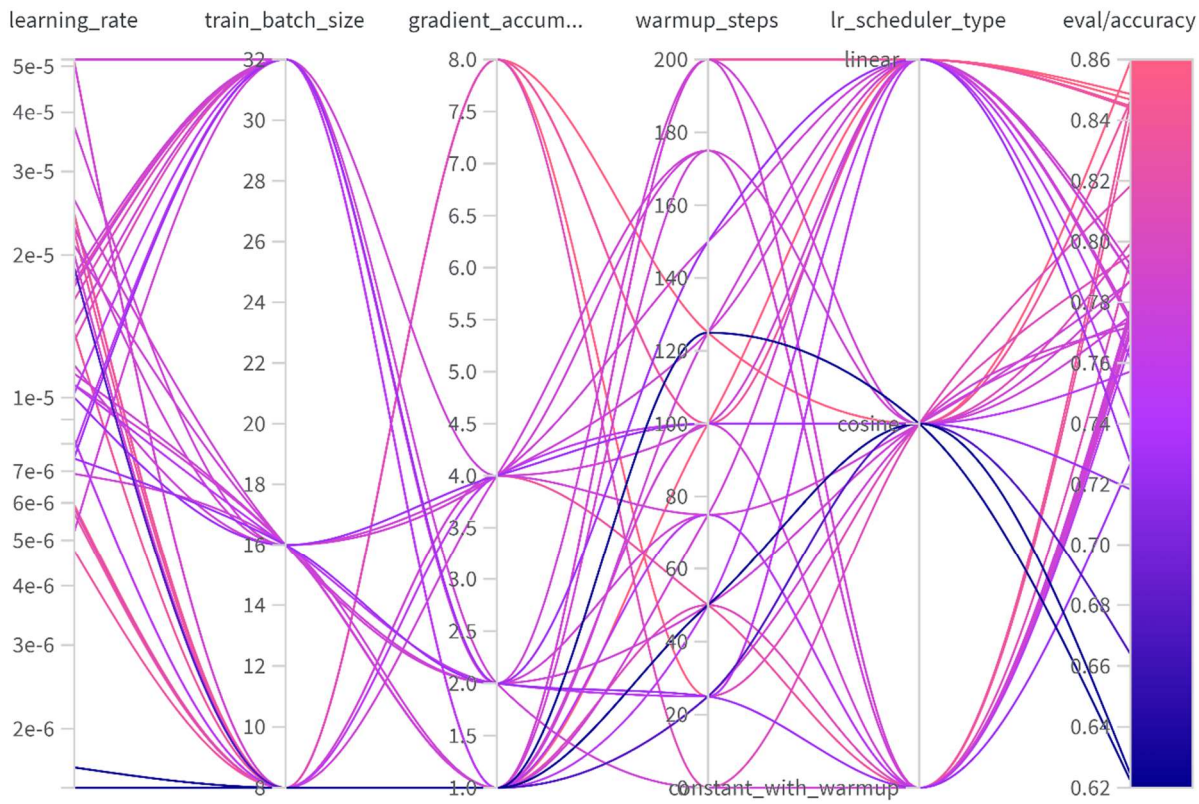
# Blackstories: ein Computer lernt Fragen zu beantworten

Maturitätsarbeit
an der Kantonsschule Uster

von

Luzian Trapletti (Jg. 2004)




Betreuerin: T. Luternauer, Expertin: Dr. G. Neyer

25. Oktober 2022

## Abstract

Diese Arbeit geht der Frage nach, ob man einen Computer als Spielpartner für das Spiel "Black-stories" benutzen kann. In diesem Spiel geht es darum, dass ein Spieler jeweils auf Basis eines kurzen Gruselgeschichten-Rätsels Fragen stellt, die vom Spielpartner nur mit Ja oder Nein beantwortet werden dürfen, bis der Spieler die Lösung herausfindet. Übersetzt in die Computerwelt handelt es sich um eine Natural Language Processing-Aufgabe mit dem Ziel, dass der implementierte Rechenalgorithmus auf beliebige Fragen des menschlichen Anwenders passend mit Ja oder Nein antwortet.

In der Arbeit wird gezeigt, dass es mittels Transferlearning möglich ist, dass ein Deep Learning Neuronales Netz in brauchbarer Zeit und ohne speziell grosse Rechenleistung so trainiert werden kann, dass der Computer tatsächlich zum nützlichen Spielpartner wird. Zum Training des Neuronalen Netzes wird der Datensatz Boolq genutzt und 3 verschiedene Datenmodelle werden bezüglich ihrer Ergebnisgüte miteinander verglichen. Nach anschliessender Hyperparameteroptimierung erreicht das Modell "Roberta Base" eine Out-of-Sample Genauigkeit von guten 78%. Dieses Modell wurde letztlich auf einem eigenen Webserver in einer selbst erstellten Webapplikation mit einem einfachen User Interface implementiert. Der Computer-Spielpartner lässt sich unter folgender Adresse ausprobieren: https://boolq.trapletti.org/ und der Source Code ist unter https://github.com/luztraplet/boolq zu finden.

# Inhaltsverzeichnis

# 1 Einleitung

## 1.1 Themeneinstieg

Künstliche Intelligenz (KI) existiert als Konzept bereits seit über 75 Jahren und spielt je länger je mehr eine wichtige Rolle in unserem Alltag [1]. Für mich war dies ein ausschlaggebender Grund mir dieses Thema für die Maturaarbeit auszusuchen. Doch was verbirgt sich hinter Künstlicher Intelligenz? KI ist ein breites Gebiet, in welchem man mit Computern versucht menschliche Intelligenz nachzuahmen. Ein Beispiel sind die hinter selbstfahrenden Autos stehenden Technologien. Maschinelles Lernen ist ein Teilgebiet der KI und befasst sich damit, Computer Aufgaben erledigen zu lassen, ohne dass sie dafür explizit programmiert worden sind. Beispielsweise wird Bilderkennung oft via Maschinellem Lernen umgesetzt [2]. Deep Learning wiederum ist ein Teilgebiet vom Maschinellen Lernen und versucht, maschinelle Intelligenz durch die Nachahmung des menschlichen Gehirns zu erreichen. Dies wird mit Neuronalen Netzen erreicht, welche wie das biologische Gehirn aus unzähligen Neuronen bestehen, die als Einheit komplexe Probleme lösen können [3].

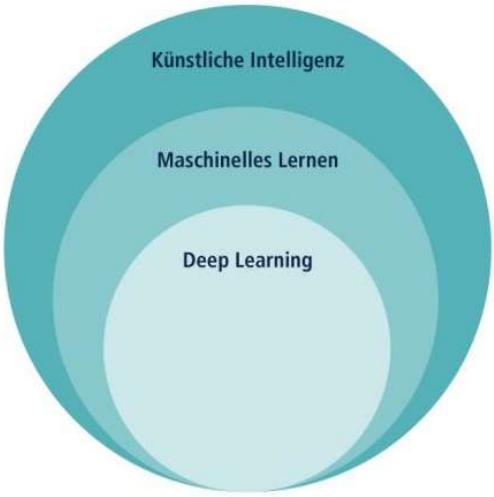



*Abbildung 1: Einordnung Deep Learning [4]*

Eine der Anwendungen des Deep Learning nennt sich Natural Language Processing (NLP). Genau dies ist Gegenstand dieser Arbeit: NLP, bei welchem es um menschliche Sprache geht, befasst sich zum Beispiel mit dem Zusammenfassen von langen Texten, dem Analysieren von Restaurant-Bewertungen oder der Beantwortung von Fragen. Das Fragen-Beantworten teilt sich auf in Ja-/Nein-Fragen und in offene Fragen. Letztere müssen mit einem Text beantwortet werden, wohingegen die Beantwortung von Ja-/Nein-Fragen eine Klassifizierung zwischen zwei Möglichkeiten (wahr oder falsch) ist.

Hier setzt diese Arbeit an: "Blackstories" ist ein Rätselspiel, welches zum Ziel hat, mittels Fragen und Ja-/Nein-Antworten Gruselgeschichten-Rätsel zu lösen: der Spieler bekommt von seinem Spielpartner eine kurze, oft morbide, Geschichte vorgelesen und muss dann mittels kurzen Fragen, die nur mit Ja oder Nein beantwortet werden dürfen, herausfinden, was in der jeweiligen Geschichte passiert ist [5].

### 1.2 Zielformulierung

Im Rahmen dieser Arbeit wird einem Computer beigebracht, wie er bei "Blackstories" als Spielpartner fungieren muss. Genauer gesagt: Es wird mittels Deep Learning eine Künstliche Intelligenz programmiert, mit welcher man das Spiel "Blackstories" spielen kann. Konkret soll letztlich ein Neuronales Netz (NN) Ja-/Nein-Fragen des Spielers beantworten können. Dies wird durch ein spezifisches Training des NN erreicht mit dem Ziel, eine bestmögliche Ja-/Nein-Fragen-Beantwortung im Zusammenhang mit der jeweiligen Blackstory zu erreichen. Schlussendlich wird eine Webapplikation entwickelt, welche es einem User erlaubt, das Spiel mit dem trainierten NN, also letztlich mit dem Computer, auszuprobieren.

# 2 Theorie

## 2.1 Deep Learning mit Neuronalen Netzen

### 2.1.1 Neuronale Netze

Deep Learning basiert auf Neuronalen Netzen (NN), welche auf dem Prinzip des Menschlichen Gehirns aufbauen, in dem (sehr) viele Neuronen miteinander verknüpft sind. Die künstlichen Neuronen sind ähnlich aufgebaut wie die biologischen Neuronen: Sie haben Eingänge (Inputs), welche Signale empfangen, sie verarbeiten diese Signale und leiten das Ergebnis anschliessend weiter (Output).

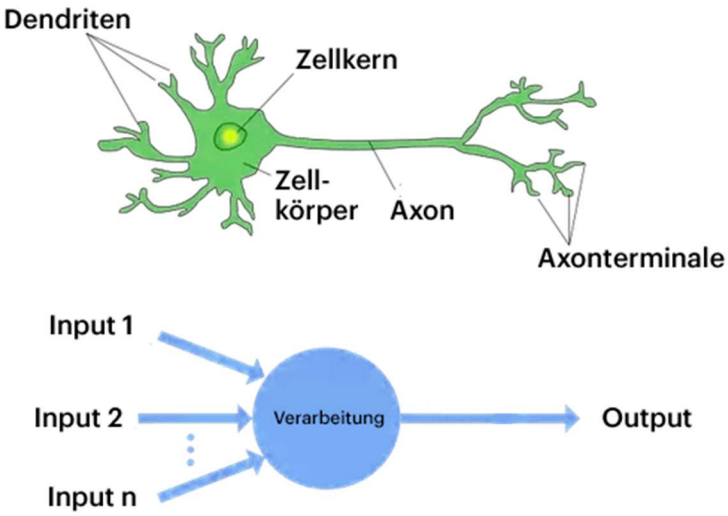



*Abbildung 2: Vergleich zwischen biologischem und künstlichem Neuron [6]*

Die Neuronen eines NN sind in Schichten (Layer) organisiert. Es gibt eine Eingabeschicht, mehrere sogenannte verborgene Schichten und einen Ausgabeschicht [3]. Jedes Neuron ist typischerweise mit jedem anderen Neuron der vorherigen und der folgenden Schicht verbunden. Das heisst, dass jedes Neuron so viele Inputs hat, wie es Neuronen in der vorhergehenden Schicht gibt und so viele Outputs, wie es Neuronen in der nachfolgenden Schicht gibt.

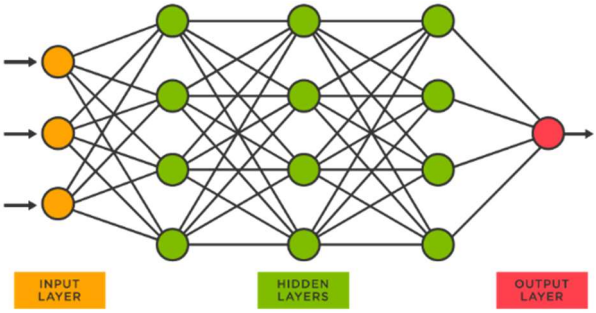



*Abbildung 3: Aufbau eines Neuronalen Netzes [7]*

Die Verarbeitung der in einem Neuron ankommenden Signale geschieht in drei Schritten: Die Werte der Eingaben werden mit einem jeweiligen Gewichts-Term multipliziert. Anschliessend wird die Summe all dieser gewichteten Eingaben mit einem Bias-Term addiert. Diese Summe wird dann durch die Aktivierungsfunktion geschickt und die Ausgabe dieser Funktion wird an die Neuronen in der nächsten Schicht weitergeleitet.

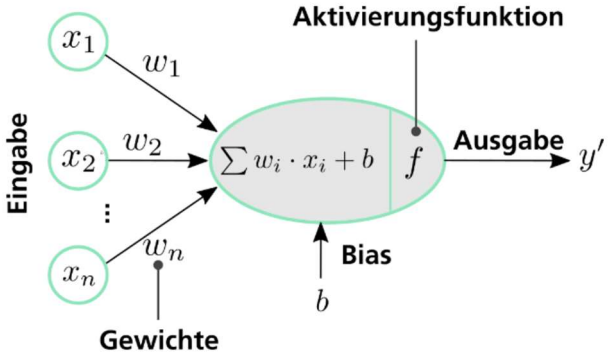



*Abbildung 4: Funktion eines Neurons [8]*

Die Gewichte und der Bias sind numerische Werte, die Aktivierungsfunktion ist eine nicht-lineare mathematische Funktion, beispielsweise der Tangens Hyperbolicus [9].

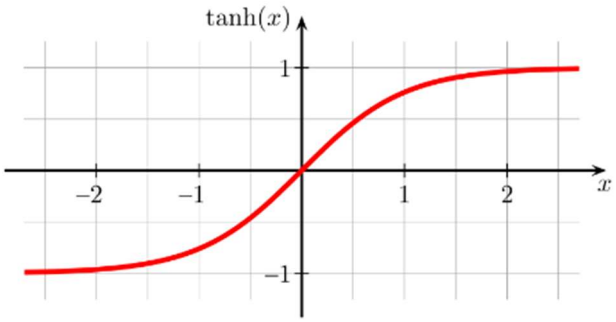



*Abbildung 5: Graph des Tangens Hyperbolicus [10]*

Auch die Eingaben müssen numerische Werte sein, können aber ganz unterschiedliche ursprüngliche Daten repräsentieren. Zum Beispiel können diese Eingaben Daten von einem Messgerät, Farbwerte eines Bildes oder auch Worte sein. Da aber Worte nicht numerisch sind, müssen diese unbedingt als Zahlenwerte codiert werden (siehe Kapitel 2.1.7), damit die oben beschriebenen Rechenschritte durchgeführt werden können.

### 2.1.2 Training

Wenn man davon spricht, dass ein Neuronales Netz lernt oder trainiert wird, meint man, dass die Parameter (alle Gewichte und alle Biases) durch sogenanntes "Trainieren" von einem Optimierer so verändert werden, dass die Ausgaben in jedem Trainingsdurchgang immer besser den gewünschten Werten entsprechen. Zum Beispiel soll ein Modell zwischen einer Katze und einem Hund unterscheiden können. Das Training bewirkt, dass das Modell nicht mehr nur mit 50% Wahrscheinlichkeit richtig auswählt, sondern beispielsweise in 70% der Fälle richtig liegt. Das Modell kann diese Aufgabe nur bewältigen, wenn es quasi "gelernt" hat, was ein Hund und was eine Katze ist. Diese Informationen sind in einem Trainings-Datensatz enthalten. Bei diesem Beispiel würde der Datensatz sinnvollerweise aus beschrifteten Bildern von Katzen und Hunden bestehen.
Ein Datensatz besteht aus einzelnen Beispielen (Samples), möglicherweise Tausenden bis Millionen, die dem Netz für das Training zur Verfügung stehen. Meist werden dem Modell mehrere Beispiele gleichzeitig präsentiert. Eine solche Einheit heisst "Batch". Die Batchgrösse beschreibt, wie viele Beispiele in einem Batch enthalten sind. Nachdem das Modell den Batch durchgerechnet hat, werden die berechneten Ausgaben mit den im Datensatz vorgegebenen Ausgaben verglichen. Im nächsten Schritt rechnet der Optimierer den Batch noch einmal durch, dieses Mal aber ausgehend von den Lösungen, also rückwärts. Dies nennt sich "Backpropagation". Dann vergleicht und optimiert er die Parameter des Modells mit dem Gradientenabstiegsverfahren, welches mit Hilfe des Differentials die bestmöglichen Veränderungen versucht herauszufinden.
Dieses Prozedere wird mit allen Beispielen des Datensatzes durchgearbeitet. Dies nennt man eine "Epoche". Oft werden sogar mehrere Epochen aneinandergehängt, um die Genauigkeit des Modells noch weiter zu steigern. Das heisst, dass jedes Sample nicht nur einmal durchgerechnet wird, sondern mehrfach.
All diese Verfahrensschritte und die dazu notwendigen Definitionen bringen Möglichkeiten aber auch Probleme mit sich: so kann es sein, dass man bei einer schlechten Definition der Trainings-Parameter den Lernprozess sogar behindert (siehe Kapitel 2.1.5) [11].

### 2.1.3 Performance

Die Genauigkeit misst, wie gut ein Modell ist. Sie ist eine Zahl zwischen 0 und 1 und besagt, wie viel Prozent der Beispiele richtig vorhergesagt werden [12].

Eine weitere wichtige Metrik ist der "Loss" eines Modells, welcher für die Verbesserung des Modells steht und möglichst klein sein sollte [13]. Um die Modell-Optimierung beeinflussen zu können gibt es Hyperparameter. Diese bestimmen zum Beispiel, wie schnell das Gradientenabstiegsverfahren vorgeht (Lernrate), welche Aktivierungsfunktion verwendet wird oder wie gross ein Batch sein soll.

### 2.1.4 In-Sample / Out-of-Sample

Um einen besseren Überblick zu haben wie gut das Modell ist, sollte man das Modell auch mit Daten testen, die während des Trainings nicht zur Verfügung gestanden haben. Solche Daten heissen Testdaten und werden auch als "Out-of-Sample" Daten bezeichnet. Hingegen jene Daten, welche für das Training verwendet wurden, heissen Trainingsdaten und werden als "In-Sample" Daten bezeichnet. Das Ziel ist, dass ein Modell eine möglichst gute Performance auf jenen Daten erreicht, auf denen es nicht spezifisch trainiert wurde. Beim Training muss man also darauf achten, dass die Performance des Modells auch bei der Nutzung von Out-of-Sample-Daten gut ist. Dies erreicht man durch eine periodische Evaluierung der Metriken (Genauigkeit und Loss) in den Testdaten [14].

### 2.1.5 Overfitting

Ein Problem Neuronaler Netze (und auch Künstlicher Intelligenz im Allgemeinen) ist das "Overfitting". Dieser Begriff beschreibt, dass sich ein Modell zu genau an den Trainingsdatensatz anpasst, was zu schlechter Genauigkeit in Out-of-Sample-Daten resultiert. Man kann sich das so vorstellen, dass das Modell nicht nur die Muster im Datensatz lernt, sondern einzelne Beispiele auswendig lernt. Dadurch wird die Fähigkeit des Modells auf Out-of-Sample-Daten zu generalisieren eingeschränkt [14]. Grundlegend wichtig ist, dass der ausgewählte Datensatz für die gewünschte Anwendung passt, da das Modell nur lernen kann, was auch im Datensatz vorhanden ist.
Wenn man ein Modell entwickelt, welches z.B. entscheiden soll, ob eine Restaurant-Bewertung positiv oder negativ ist, nutzt es nichts, wenn man das Modell mit einem Übersetzungsdatensatz trainiert hat: der Datensatz muss so exakt wie möglich mit der gewünschten Anwendung übereinstimmen [15]. Konkret werden die Modelle mit einem allgemeinen Datensatz zur Beantwortung von Ja-/Nein-Fragen trainiert. Dies erlaubt es, die Modelle anschliessend für beliebige Blackstories zu verwenden.

### 2.1.6 Transferlernen

Eine weitere Technik im Kontext von NN, das Transferlernen, benutzt ein Basismodell, welches bereits auf einem Datensatz trainiert wurde. Anschliessend wird dasselbe Modell auf einem weiteren domänenspezifischen Datensatz trainiert (quasi ein "Finetuning"). Die Idee dahinter ist, dass eine Entwickler nicht ein komplett neues Modell trainieren muss, sondern bereits auf einem vorhandenen Modell aufbauen kann [16]. Dadurch spart man extrem viel Rechenpower und Zeit (zum Beispiel wurde das Modell Roberta auf 160GB Text für einen ganzen Tag auf 1024 V100 GPUs trainiert [17, 18]; jede dieser GPUs kostet ca. CHF 10'000.- (!) [19]). Dies macht deutlich: Transferlernen ist unabdingbar, will man in absehbarer Zeit oder mit beschränkten finanziellen Mitteln zu einem performanten Modell kommen. Das "Finetuning" bzw. Transferlernen ist bei weitem weniger aufwendig und führt innerhalb weniger Stunden bereits zu guten Ergebnissen. Ein Nachteil von Transferlernen ist, dass man zum Beispiel bei Anwendungen im Bereich der menschlichen Sprache eingeschränkt ist: Viele der grossen Modelle sind auf Englisch trainiert worden. Daher muss das Endmodell auch auf Englisch sein. Das Gleiche gilt für digitale Datensätze (Unmengen an Trainingsdaten, die im Internet zur Verfügung stehen), welche meist Englisch nutzen. Aus diesem Grund musste die ganze Anwendung auf Englisch realisiert werden.

### 2.1.7 Vorverarbeitung (Preprocessing)

Wie oben erklärt rechnet ein Neuronales Netz mit Zahlen. Im Bereich des NLP muss man vor dem Training die Worte in eine numerische Repräsentation übersetzen. Dies übernimmt der sogenannte "Tokenizer". Er ist eigentlich ein grosses Vokabelbuch, das jedem Wort, zum Teil auch gewissen Silben, eine einzigartige Zahl zuweist. Um also ein Beispiel oder einen ganzen Batch durch das Modell laufen zu lassen, muss der Tokenizer die Worte zuerst in Zahlen umwandeln, welche dann durch das Modell geschickt werden können [20].

### 2.1.8 Nachbearbeitung (Postprocessing)

Die Ausgaben eines Modells sind Zahlen, wobei die Bedeutung dieser Zahl abhängig von der Anwendung ist. Zum Beispiel könnten es numerische Repräsentationen von Worten sein, die erneut durch den Tokenizer geschickt werden müssen, um sie für uns lesbar zu machen (dies gilt für die Beantwortung offener Fragen). Da im Fall der Blackstories eine Klassifizierung stattfindet (richtig, falsch), benötigen wir nachfolgend bei der Interpretation keinen Tokenizer.

Die Ausgabe der Modelle, die für die Problemstellung trainiert wurden, ist ein Vektor mit zwei Komponenten. Diese zwei Zahlen stehen für die zwei möglichen Zustände (Richtig und Falsch) und können nach einer Softmax-Transformation als Wahrscheinlichkeiten interpretiert werden. Die Softmax-Transformation bewirkt, dass die Summe aller Einträge gleich eins ist und erlaubt somit die Interpretation als Wahrscheinlichkeiten [21]. Wenn man jedoch nur wissen will, welcher der Zustände der Wahrscheinlichste ist, kann man einfach jenen mit der grössten Zahl/Wahrscheinlichkeit nehmen. Diese Transformation nennt sich Argmax [22].

### 2.1.9 Hyperparametertraining

Ein wichtiges Tool für das Verbessern eines Modells ist das Hyperparametertraining, auch Hyperparametersuche genannt. Bei diesem Vorgehen wird das Modell mehrere Male trainiert, wobei jeweils andere Hyperparameter verwendet werden. Zum Beispiel wird das Modell zehn Mal trainiert, jedes Mal mit einer leicht anderen Lernrate. Beim Vergleich der End-Genauigkeiten wird auffallen, dass das Modell unterschiedlich gut gelernt hat. Wenn man dieses Verfahren systematisch anwendet, sollte man bessere Ergebnisse erhalten, als wenn man die Hyperparameter "von Hand" schätzt [23]. Für eine effektive Optimierung ist es sinnvoll, sich auf gewisse wichtige Hyperparameter zu beschränken, da sonst extrem viel Zeit und Rechenpower notwendig wird. Im Rahmen dieser Arbeit wurde eine Beschränkung auf die folgenden Hyperparameter vorgenommen: Lernrate, Batchgrösse, Gradienten-Akkumulationsschritte, Aufwärmschritte, Lernraten Planer und Anzahl Epochen.

- Die **Lernrate** ist mitverantwortlich für das Verhalten des Gradientenabstiegsverfahren. Dieses versucht mit Hilfe des Differentials eine Verbesserung des Modells zu erreichen, welche mit der Lernrate beschleunigt werden kann. Je grösser der Wert ist, desto wahrscheinlicher wird es, dass das Verfahren zu schnell voranschreitet und mögliche Genauigkeits-Optima/Loss-Minima nicht erreicht. Zu kleine Werte hingegen führen zu einem langsamen Vorgehen des Verfahrens [24]. Abbildung 6 stellt den Einfluss der Lernrate auf das Verfahren der Loss-Minimierung durch den Gradientenabstiegsalgorithmus dar.

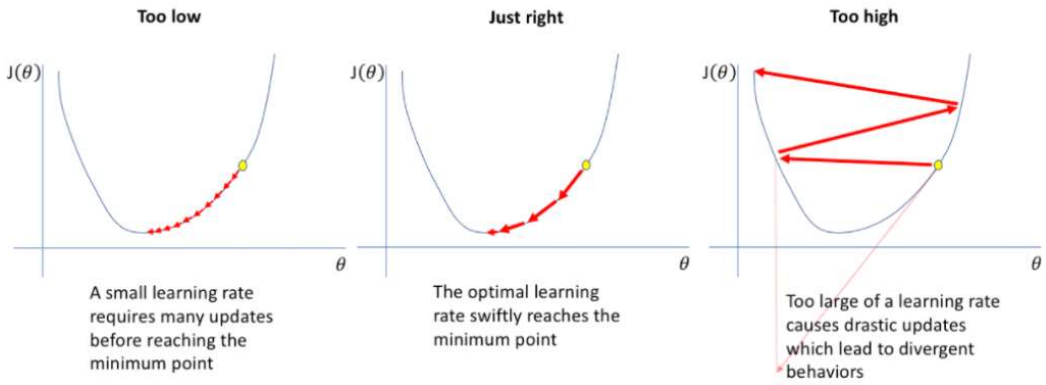



*Abbildung 6: Einfluss der Lernrate auf Gradientenabstiegsverfahren [25]*

- Die **Batchgrösse** gibt an, wie gross ein Batch ist. Wie oben genannt, landen alle Samples aus einem Batch gemeinsam im Optimierer. Dies führt dazu, dass das Modell nicht nur für Muster aus einem Sample optimiert wird, sondern für Muster, die im ganzen Batch vorhanden sind. Je grösser der Batch ist, desto wahrscheinlicher ist es, dass die Muster im Batch auch Muster aus dem ganzen Datensatz sind. Zu vergleichen ist dies mit dem Lernen einer grammatikalischen Regel. Man kann nur auf die Regel schliessen, falls man genug Beispiele hat. Ansonsten kann man sich nicht sicher sein, dass das entdeckte Muster wirklich die generelle Regel zeigt. Hingegen überfüllen zu grosse Batches den Arbeitsspeicher, da alle Samples des Batches dort gleichzeitig geladen sein müssen [26].
- Die **Gradienten-Akkumulationsschritte** sollen das Problem eines zu grossen Batches lösen. Man wählt die Batchgrösse nur so gross, dass der Computerspeicher nicht überschritten wird und rechnet dafür mehrere Batches hintereinander ohne Optimieren durch und optimiert erst danach alle gemeinsam. Die effektive Batchgrösse ist also ein Vielfaches des unter "Batchgrösse" angegebenen Wertes [27].
- Die Anzahl der **Aufwärmschritte** entspricht der Anzahl Beispiele, die optimiert werden, bis die effektive Lernrate auf dem angegebenen Wert ist. Das heisst, dass die effektive Lernrate während eines Trainings langsam über die spezifizierte Anzahl Beispiele von Null bis auf den angegebenen Wert linear erhöht wird. Wenn man keine Aufwärm-Periode haben will, setzt man die Anzahl Aufwärmschritte auf Null, dann ist die effektive Lernrate von Beginn an gleich der angegebenen Lernrate [27].

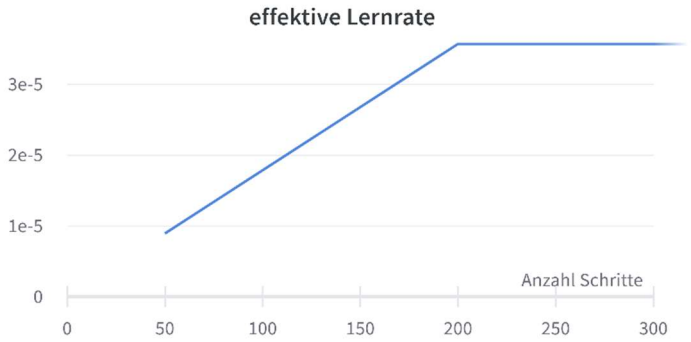



*Abbildung 7: Effektive Lernrate während Aufwärmperiode mit 200 Aufwärmschritten*

- Durch den **Lernraten Planer** wird die effektive Lernrate nach der Aufwärmperiode erneut gesenkt. Verschiedene Planer senken die effektive Lernrate auf unterschiedliche Weise. Häufige Varianten sind lineare Planer oder kosinusartige, es gibt aber auch konstante, welche die effektive Lernrate auf dem angegebenen Wert belassen [27]. Folgende Grafik stellt die effektive Lernrate eines kosinusartigen Lernraten Planers mit kurzer Aufwärm-Periode dar.

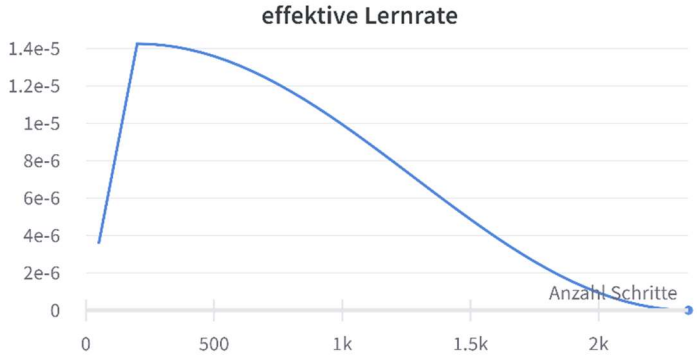



*Abbildung 8: Effektive Lernrate während ganzem Training*

- Die **Anzahl Epochen** spezifiziert, wie oft der ganze Datensatz durchgearbeitet wird. Zu viele Wiederholungen führen üblicherweise zu einem Overfitting, welches nicht erwünscht ist. Zu wenige Wiederholungen können bedeuten, dass das Modell noch nicht alles aus sich und aus dem Datensatz herausholen konnte. Das Potenzial wurde also nicht voll ausgeschöpft [27].

Wichtig anzumerken ist, dass ein Teil der Parameter des vortrainierten Modells, oft die letzte Schicht, beim Initialisieren (Laden) nicht übernommen wird und mit zufälligen Werten ersetzt wird. Diese Zufallszahlen kann man beeinflussen und mit einem sog. "Seed" reproduzierbar machen. Der Seed ist eine Zahl, welche einen Zufallsgenerator immer die gleichen (pseudozufälligen) Zahlen produzieren lässt. Der Seed hat einen erstaunlich grossen Einfluss auf die Performance eines Modells [28]. Auch diesen Seed könnte man daher beim Hyperparametertraining optimieren, dies wird aber im Rahmen dieser Arbeit nicht durchgeführt.

#### 2.1.10 Hardware

Die Dauer eines Trainings hängt von der Grösse des Modells ab und wie viele Epochen gerechnet werden. Hinzu kommt noch der Hardware-Aspekt: für ein Training kommen drei Möglichkeiten in Frage: CPU (central processing unit), GPU (graphics processing unit) und TPU (tensor processing unit). Eine oder mehrere CPUs stehen in jedem Computer zur Verfügung, sind jedoch vergleichsweise langsam für ein solches Training. Auch GPUs sind mittlerweile in fast jedem Computer verbaut, jedoch gibt es grosse Unterschiede: von einer unmerklichen Beschleunigung des Trainings bis hin zu einer Verxfachung der Geschwindigkeit. TPUs hingegen finden sich nur in speziell dafür konzipierter Hardware und bieten eine enorme Beschleunigung an. Google bietet mit dem Google Colab Space eine Cloud Computing Plattform an, mit welcher man auf diverse Hardware zugreifen kann. Bei der Gratisversion hat man begrenzte Nutzungszeiten von GPUs und kann TPUs nicht richtig nutzen. Mit einem sogenannten Pro Account kann man jedoch die angebotenen GPUs und TPUs voll benutzen, wobei man dazu allerdings zuerst einige notwendige Umstrukturierungen auf Programmebene vornehmen muss [29].

### 2.2 Webapplikation

#### 2.2.1 Architektur

Eine Webapplikation besteht aus einem Frontend und einem Backend. Das Frontend wird auch User-Interface (UI) genannt. Alles, was der Besucher einer Webseite sieht, ist Teil des UIs. Sogenannte Callbacks verbinden UI-Elemente, wie zum Beispiel Eingabefelder, mit dem Backend der Applikation. Das Backend stellt alle Funktionalitäten im Hintergrund zur Verfügung, welche vom Nutzer nicht direkt ersichtlich sind. Im Rahmen dieser Arbeit ist der gesamte Bereich des NN Teil des Backends.

Das UI der Webseite ist wie ein Chat-UI aufgebaut, so dass der Benutzer Ja-/Nein-Fragen eingeben kann und das Modell anschliessend mit Ja oder Nein antwortet. Damit das Spiel ein Ende hat, muss ein sekundäres Modell in die Applikation integriert werden, welches kontrolliert, ob der Spieler die Lösung des Rätsels erraten hat. In diesem Falle wird das zweite Modell dem UI bzw. dem Spieler mitteilen, dass das Rätsel gelöst wurde, der Spieler also die Lösung erraten hat. Dieses sekundäre Modell ist von "Huggingface" und kann unter folgendem Link eingesehen werden: https://huggingface.co/Prompsit/paraphrase-bert-en

### 2.2.2 Deployment

Für das Deployment (Zurverfügungstellung) einer Webapplikation kommen verschiedene Optionen in Frage. Zum Beispiel hätte man die Webapplikation auf einer spezialisierten Cloudplattform wie "Heroku", welche Python Webapplikationen sehr einfach hosten kann, deployen können. Da Machine Learning-Apps allerdings performante Hardware brauchen, wäre diese Option auf Heroku sehr teuer geworden. Als Alternative wurde deshalb ein eigener Cloud-Server aufgesetzt, der im Vergleich zu beispielsweise Heroku wesentlich bessere Hardware für den gleichen Preis bietet. Bei einer eigenen Lösung muss man sich allerdings selbst um wichtige Server-Aspekte, wie Security, Routing, Crash-Erkennung und Wartung kümmern. Kommerzielle Anbieter wie Heroku übernehmen diese Aufgaben und kosten daher am Ende ein Vielfaches mehr als ein gleich performanter, selbst aufgesetzter Server.

# 3 Implementierung

## 3.1 Umsetzung Neuronales Netz

### 3.1.1 Entwicklungsumgebung

Das NN wurde mit Python 3.10.5, mit dem Code Editor IntelliJ IDEA Community Edition und Google Colab entwickelt. Als Python Pakete wurden Pytorch [30] und Huggingface [31] für das Entwickeln des NN verwendet. Pytorch ist eines der führenden Pakete im Bereich Deep Learning, welches gut mit Huggingface integriert ist. Huggingface bietet viele Dienstleistungen im Bereich NN an und kann zum Beispiel für das Laden von vortrainierten NN und Datensätzen gebraucht werden. Ausserdem bietet Huggingface Trainingsalgorithmen an, welche ganze Trainings durchführen können.
Unter den Datensätzen von Huggingface ist BoolQ [32], welcher für das Training der Modelle verwendet wurde. BoolQ ist ein englischer Datensatz zur Beantwortung von Ja-/Nein-Fragen. Die meisten Datensätze sind mindestens in zwei Teile geteilt: Der Teil "Training" ist für das Trainieren bestimmt, der Teil "Test"/"Evaluation" ist für die Ermittlung von Out-of-Sample Ergebnissen bestimmt. Im Fall von BoolQ enthalten beide Teile zusammen rund 12'000 Beispiele. Jedes Beispiel besteht aus einer Frage, einem kurzen Text und einem Label (Lösung). Die Frage ist eine Ja-/Nein-Frage, welche mit den Informationen aus dem kurzen Text beantwortet werden kann. Das Label gibt an, ob die Frage mit Ja oder Nein beantwortet werden sollte. Inhaltlich/thematisch unterscheiden sich alle Beispiele voneinander [33].
Mit dem API von Huggingface kann man den Datensatz sowie den Tokenizer und das vortrainierte Modell herunterladen. Mit dem von Huggingface bereitgestellten Trainer kann man die Hyperparameter des Trainings spezifizieren und dann ein Training durchführen. Der Trainer kann auch selbst entwickelt werden. Dies bietet einige Vorteile, da man mehr Einfluss über gewisse Teile des Trainings hat. Dies wurde anfänglich ausprobiert, im weiteren Verlauf wurden jedoch mehrheitlich vordefinierte Trainer von Huggingface verwendet.
Die Integration in den Google Colab Space ist vergleichsweise einfach. Sobald man aber nicht mehr nur mit CPUs arbeitet, sondern auch GPUs und TPUs nutzen will, muss man gewisse Aspekte im Trainer verändern und um auf TPUs arbeiten zu können, müssen zusätzliche Schritte vorgenommen werden. Die Nutzung von TPUs wurde ausprobiert. Jedoch ist die Verwendung von GPUs einfacher und daher wurden nur diese für die endgültigen Trainings verwendet.

Ein weiteres hilfreiches Tool ist "Wandb" (Weights and Biases) [34], welches das Tracking von Trainings einfach macht. Der Trainer leitet den Stand des Trainings automatisch an Wandb weiter, in welchem man anschliessend Zusammenfassungen und weitere Informationen auf Basis der gesammelten Daten analysieren kann.
Um das Transferlernen durchzuführen, muss vor dem Training ein geeignetes Basismodell geladen werden. Modelle, die für das Klassifizieren von Texten ausgelegt sind, heissen Autoencoder [35]. Zwei Beispiele davon sind das Bert-Modell [36] und das Modell Roberta (Weiterentwicklung von Bert) [17].
Das Speichern eines feinabgestimmten Modells kann entweder lokal oder auf Huggingface geschehen. Dadurch muss man beim erneuten Verwenden des Modells nicht mehr das ganze Training durchführen, sondern kann einfach auf das fertige Modell zugreifen.

### 3.1.2 Implementierung des Trainings

Der Anfang eines Trainingsskripts sieht meist ähnlich aus, da immer zuerst Boolq von Huggingface geholt wird, welches dann mit dem importierten Tokenizer vorverarbeitet wird. Anschliessend wird das Basismodell, zum Beispiel Roberta, importiert, sowie der passende Optimierer aufgesetzt. Dann werden entweder die Trainingsargumente für den Trainer von Huggingface spezifiziert oder eine eigene Trainingsschleife aufgesetzt. Wenn man den Trainer von Huggingface benutzt, müssen die Metriken aufgesetzt werden, anschliessend kann das Training gestartet werden. In der eigenen Trainingsschleife passiert ganz vereinfacht folgendes: Jedes Beispiel wird zuerst durch das Modell geschickt, durch die Backpropagation der Ausgaben werden dann die bestmöglichen Veränderungen der Parameter des Modells ausgerechnet, welche anschliessend durch den Optimierer umgesetzt werden.
Um die Performance eines Modells zu steigern, muss man die richtigen Hyperparameter in den Trainingsargumenten spezifizieren. Will man sich nicht nur auf sein Bauchgefühl verlassen, kann man den Hyperparameteroptimierer von Huggingface verwenden. Das Hyperparameter-API von Huggingface beruht auf Raytune [23], welches Hyperparametertrainings mit verschiedenen Algorithmen anbietet. Bei der Anwendung muss man angeben, welche Hyperparameter der Algorithmus optimieren soll und welche Werte diese annehmen dürfen. Zum Beispiel muss man spezifizieren, dass die Anzahl Epochen zwischen 2 und 5 liegen darf und die Lernrate zwischen $10^{-3}$ und $10^{-6}$. Der Algorithmus startet dann hintereinander mehrere Trainings mit unterschiedlichen Hyperparametern und leitet die Ergebnisse an Wandb weiter, welches zur Interpretation der Ergebnisse verwendet werden kann.

Beim Code eines Hyperparametertrainings kann man viel von einem normalen Training mit dem Huggingface Trainer übernehmen, muss aber zusätzlich den Suchbereich der einzelnen zu optimierenden Parametern angeben und am Schluss kein normales Training aufrufen, sondern den Suchalgorithmus von Raytune.
Um das bestmögliche Modell zu finden, müssen verschiedene Modelle trainiert werden, sodass das Potenzial der Modelle abgeschätzt und vergleichen werden kann. Anschliessend kann man jenes der Modelle, welches als bester Kandidat identifiziert wurde, mit einem Hyperparametertraining weiter optimieren. Zuletzt wird ein weiteres Training mit den gefundenen Hyperparametern durchgeführt und das Modell anschliessend gespeichert.

## 3.2 Umsetzung der Webapplikation

Die Einbettung des NN in eine Webapplikation wurde ebenfalls in Python implementiert. Für die Entwicklung des UI wurde Dash Plotly [37] verwendet. Das mit Dash entwickelte UI kann mit den Backend-Funktionen via Callbacks kommunizieren. Callbacks sind Funktionen, die z.B. mit einem UI-Element, wie dem Submit Button, verknüpft werden können. Falls der Submit Button vom Spieler gedrückt wird, wird die entsprechende Callback Funktion aufgerufen, welche danach weitere Funktionen aufrufen kann.
Das Backend wurde entworfen ohne auf die Modelle von Huggingface zugreifen zu müssen, da solche Online-Anfragen die App sonst merklich verlangsamt hätten. Die Modelle wurden deshalb lokal im Applikationscode gespeichert, um von dort während des Starts der App in den Speicher geladen zu werden. Für das UI wurde zusätzlich Dash Bootstrap [38] und CSS [39] verwendet. Mit Dash Plotly und Dash Bootstrap kann das UI ähnlich wie eine HTML-Seiten entworfen werden. Zusätzlich wurde das UI so angelegt, dass es auch auf kleineren Bildschirmen, wie Handys gut passt.
Das Hosting der Webseite wurde auf einem Ubuntu Server von Vultr [40] gemacht. Weiter wurde ein automatischer Restart angelegt, welcher die Applikation bei allfälligen Abstürzen automatisch neu startet. Zusätzlich wurden SSL-Zertifikate [41] für die Webseite eingerichtet, so dass diese mittels sicherer Https-Verbindung erreicht werden kann. Weiter wurde für die Webseite eine eigene Adresse eingerichtet und die entsprechenden DNS-Server konfiguriert, welche die IP-Adresse des Servers mit der Adresse verknüpfen.
Der gesamte Source-Code der Webapplikation ist unter https://github.com/luztraplet/boolq zu finden.

# 4 Ergebnisse

## 4.1 Transfertraining auf Roberta Base

Im Folgenden wird beispielhaft ein Transfertraining eines Roberta-Base Modells vorgestellt. Das Modell besitzt 12 verborgene Schichten mit insgesamt 125 Millionen trainierbaren Parametern. Dies führt zu einer Modellgrösse von 501 MB. Das Modell wurde mit dem Trainer von Huggingface auf dem Datensatz BoolQ trainiert. Folgende Hyperparameter wurden vor dem Training auf die angegebenen Werte gesetzt. Alle nicht aufgeführten Hyperparameter haben Standardwerte.

| | Wert |
|---|---|
| Lernrate | $5*10^{-5}$ |
| Batchgrösse | 32 |
| Gradienten-Akkumulationsschritte | 1 |
| Lernratenplaner | Linear |
| Anzahl Aufwärmschritte | 50 |
| Anzahl Epochen | 5 |

*Tabelle 1: Werte der Hyperparameter im Beispieltraining*

Das Training wurde in einem Google Colab mit Zugriff auf eine Tesla K80 GPU in ungefähr 150 Minuten durchgeführt. Total wurden $6*10^{15}$ Berechnung durchgeführt. Der folgende Verlauf der Testgenauigkeit und des Loss wurde durch periodische Evaluierung bestimmt.

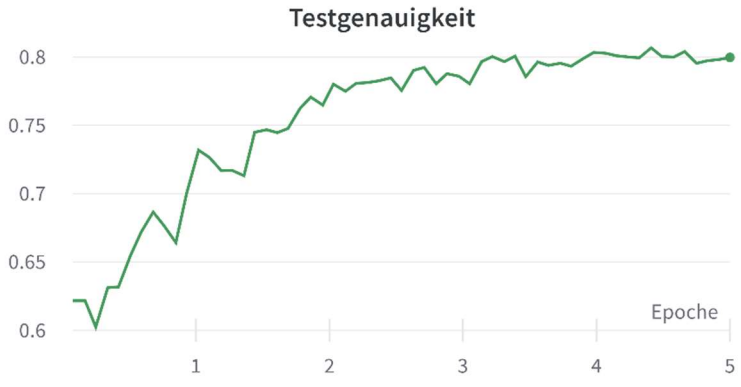



*Abbildung 9: Verlauf der Testgenauigkeit*

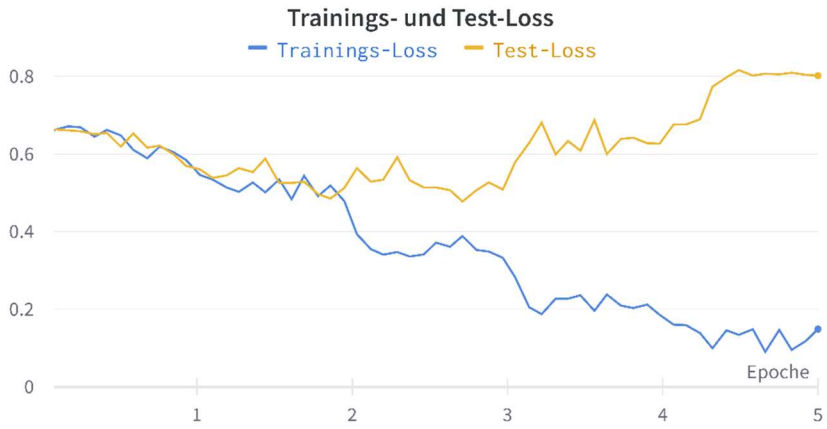



*Abbildung 10: Verlauf des Trainings- und Test-Loss*

Die X-Achse gibt jeweils in Abb. 9 und 10 an, wie viele Epochen bis zum Datenpunkt durchgeführt wurden, die Y-Achse bei Abb. 9 entspricht der Genauigkeit und bei Abb. 10 dem Loss. Die Genauigkeit des Modells steigt in den ersten zwei Epochen von 60% auf 78% stark an, flacht danach jedoch ab und erreicht am Schluss knapp 80%. In Abb. 10 sinken Trainings- und Test-Loss in den ersten zwei Epochen, wobei der Test-Loss danach ansteigt. Der Trainings-Loss sinkt weiter bis zum Ende des Trainings. Das Auseinanderdriften des Test- und Trainings-Loss lässt auf ein Overfitting schliessen, welches auch in Abb. 9 durch das Abflachen des Graphen zu erahnen ist. Um dieses Netz für eine Anwendung zu verwenden, müsste man das Training noch vor dem Einsetzten des Overfittings stoppen, also ca. Ende zweiter Epoche, was einer Out-of-Sample Genauigkeit von ca. 78% entspräche. Verglichen mit den Einträgen der BoolQ Benchmark (Rangliste der besten Modelle) ist diese Performace knapp ausserhalb der Top 10 anzusiedeln, wobei die besten Modelle eine Genauigkeit von über 92% erreichen [42]. Dieser Vergleich ist jedoch nicht ganz fair, da diese Modelle mit bis zu 270 Milliarden Parametern über 2'000 Mal grösser sind als das hier genutzte Modell. Im Vergleich dazu erreicht der Mensch eine Testgenauigkeit von 90% [32].

### 4.2 Vergleich verschiedener Modelle

Im Folgenden werden die Trainings dreier Modelle verglichen: Das oben vorgestellte Modell Roberta Base (125 Mio. Parameter), die grössere Variante Roberta Large (355 Mio. Parameter) und ein kleineres Modell namens Albert Base (11 Mio. Parameter). Folgende Hyperparameter wurden verwendet:

| | Roberta Base | Roberta Large | Albert Base |
|---|---|---|---|
| Anzahl Epochen | 5 | 5 | 5 |
| Lernrate | $5*10^{-5}$ | $10^{-5}$ | $10^{-5}$ |
| Batchgrösse | 32 | 8 | 32 |
| Gradienten-Akkumulationsschritte | 1 | 4 | 1 |
| Lernraten Planer | Linear | Linear | Linear |
| Anzahl Aufwärmschritte | 50 | 50 | 50 |

*Tabelle 2: Werte der Hyperparameter für 3 Modelle*

Die Testgenauigkeiten sowie der Trainings- und Test-Loss der drei Modelle sehen nach einem identischen Training wie in 4.1 wie folgt aus:

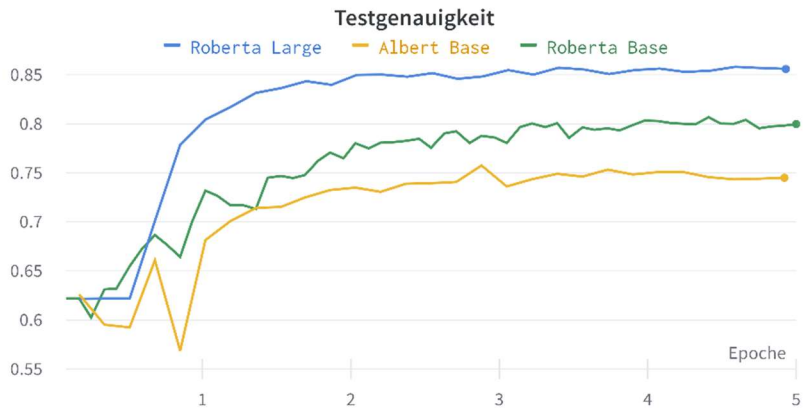



*Abbildung 11: Verlauf der Testgenauigkeiten der drei Modelle*

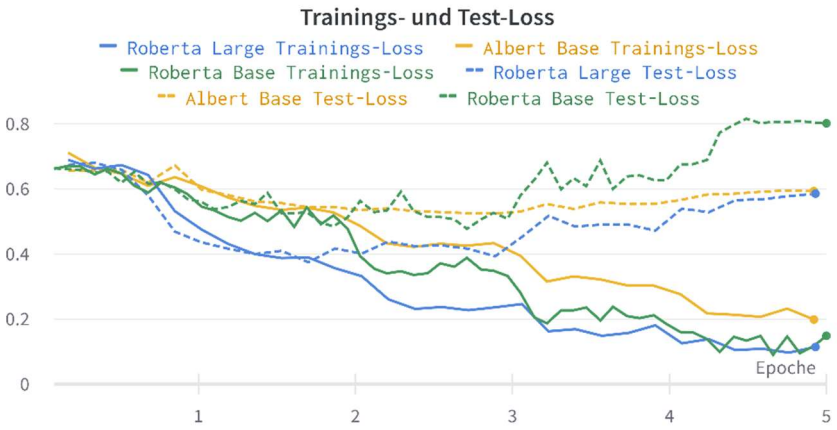



*Abbildung 12: Verlauf des Trainings- und Test-Loss der drei Modelle*

Die X-Achse gibt jeweils an, wie viele Epochen bis zum Datenpunkt durchgeführt wurden, die Y-Achse bei Abb. 11 entspricht der Genauigkeit und bei Abb. 12 dem Loss. Bei allen Modellen hat erneut nach ungefähr zwei Epochen ein Overfitting eingesetzt, welches durch das Auseinanderdriften der beiden Loss Metriken zu erkennen ist. Der Vergleich der Genauigkeiten der Modelle am Ende der zweiten Epoche zeigt, dass das grösste Modell Roberta Large mit 84% um einiges besser ist als das zweitgrösste Modell Roberta Base. Dieses erreicht mit 78% wiederum eine signifikant bessere Genauigkeit als das kleinste Modell Albert mit 73%. Grössere Modelle sind also tendenziell besser als kleinere, jedoch dauern das Training und die Berechnungen massiv länger. Deshalb muss immer eine Abwägung zwischen Performance und Geschwindigkeit gemacht werden. Im Verlauf dieser Arbeit wurde darum das mittlere Modell Roberta Base für die weiteren Schritte ausgewählt.

### 4.3 Hyperparametertraining

Um das Modell Roberta Base zu optimieren, wurde ein Hyperparametertraining durchgeführt. Das Modell wurde 20-mal mit unterschiedlichen Hyperparametern trainiert. 12 der 20 Modelle haben in der ersten von drei Epochen praktisch nichts dazugelernt, wurden daher frühzeitig gestoppt und werden in der folgenden Analyse nicht weiter behandelt. Bei den nächsten zwei Grafiken sind alle verbliebenen 8 Läufe gruppiert dargestellt. Der eingefärbte Bereich gibt den minimal und maximal gemessenen Wert aller Trainings an. Die Linie ist jeweils der Durchschnitt aller Werte.

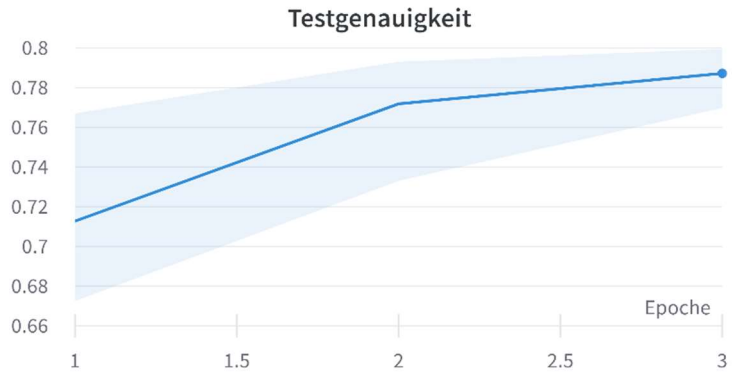



*Abbildung 13: Testgenauigkeit aller Läufe gruppiert*

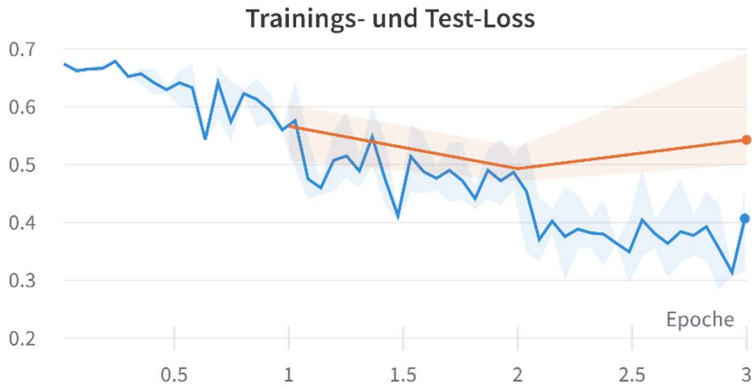



*Abbildung 14: Trainings- und Test-Loss aller Läufe gruppiert*

Erneut ist in Abb. 14 ein Overfitting aller Modelle in der dritten Epoche auszumachen. Zudem ist in Abb. 13 zu erkennen, dass das beste Training nur ganz knapp besser ist als das Training aus 4.1 mit selbst ausgewählten Hyperparametern. Der Durchschnitt aller Trainings liegt signifikant unter der Testgenauigkeit aus 4.1, obwohl die Modelle, welche fast gar nichts gelernt haben, bereits vorab ausgeschlossen worden sind. Wenn diese im Durchschnitt enthalten wären, würde es sogar noch schlechter aussehen.

In der folgenden Abbildung werden die verwendeten Hyperparameter in Relation zur erreichten Testgenauigkeit gestellt. Jede farbige Linie entspricht einem der acht Läufe. Der Schnittpunkt jeder Linie mit den einzelnen Achsen entspricht dem verwendeten Wert für den angegebenen Hyperparameter. Die Farbe der Linie ist von der endgültigen Genauigkeit des Laufes abhängig (Achse ganz rechts). Zum Beispiel hatte der Lauf mit der schlechtesten Genauigkeit (dunkelste Linie) eine Lernrate von ungefähr $5*10^{-6}$, eine Batchgrösse von 32 und Null Aufwärmschritten.

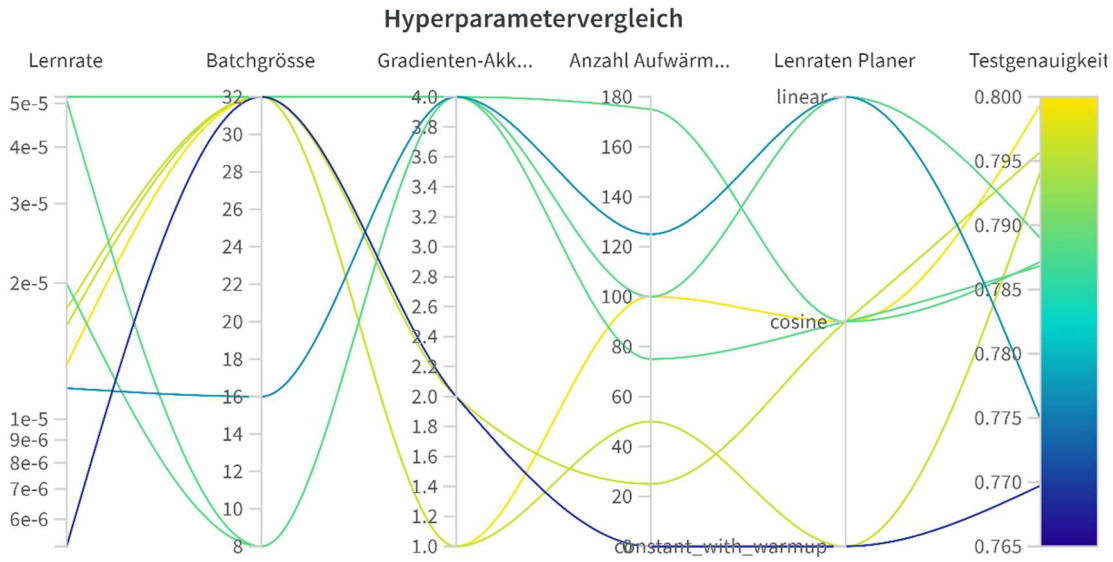



*Abbildung 15: Hyperparameter im Vergleich zur endgültigen Testgenauigkeit*

Abb. 15 gibt Aufschluss über den Einfluss der 5 Hyperparameter, welche optimiert wurden. Zum Beispiel ist zu erkennen, dass Lernraten zwischen $2*10^{-5}$ und $10^{-5}$ tendenziell zu einem besseren Modell führen. Weiter kann darauf geschlossen werden, dass hohe Aufwärmschritte zu einem schlechteren Resultat führen. All diese Erkenntnisse können für die Verbesserung der Hyperparameter genutzt werden. Wenn man mehr Varianten an Hyperparametern ausprobieren würde, könnte man vermutlich weitere Schlüsse aus dem Hyperparametertraining ziehen. Dafür wären aber mehr Zeit und Rechenpower erforderlich, welche den Rahmen dieser Arbeit gesprengt hätten.

Für das finale Modell, welches in der Webapplikation verwendet wird, wurden folgende Hyperparameter verwendet, welche zu einer Testgenauigkeit von 78.5% in 2 Epochen führten.

| | Wert |
|---|---|
| Lernrate | $3.72*10^{-5}$ |
| Batchgrösse | 16 |
| Gradienten-Akkumulationsschritte | 1 |
| Lernratenplaner | Linear |
| Anzahl Aufwärmschritte | 150 |

*Tabelle 3: Werte der Hyperparameter für finales Modell*

### 4.4 Webseite

Die Webseite ist unter https://boolq.trapletti.org/ zu finden. Beim Öffnen der Webseite erscheint zuerst ein Fenster mit "Terms and Conditions". Diesen muss man zustimmen. Grund für die „Terms and Conditions“ ist, dass die Webseite öffentlich zugänglich ist und dass es bei NN im Zusammenhang mit NLP eine kleine Wahrscheinlichkeit für nicht erwünschtes/korrektes Verhalten gibt (z.B. können NLP-Modelle unter bestimmten Umständen rassistische Antworten geben). Anschliessend ist die Hauptseite sichtbar:

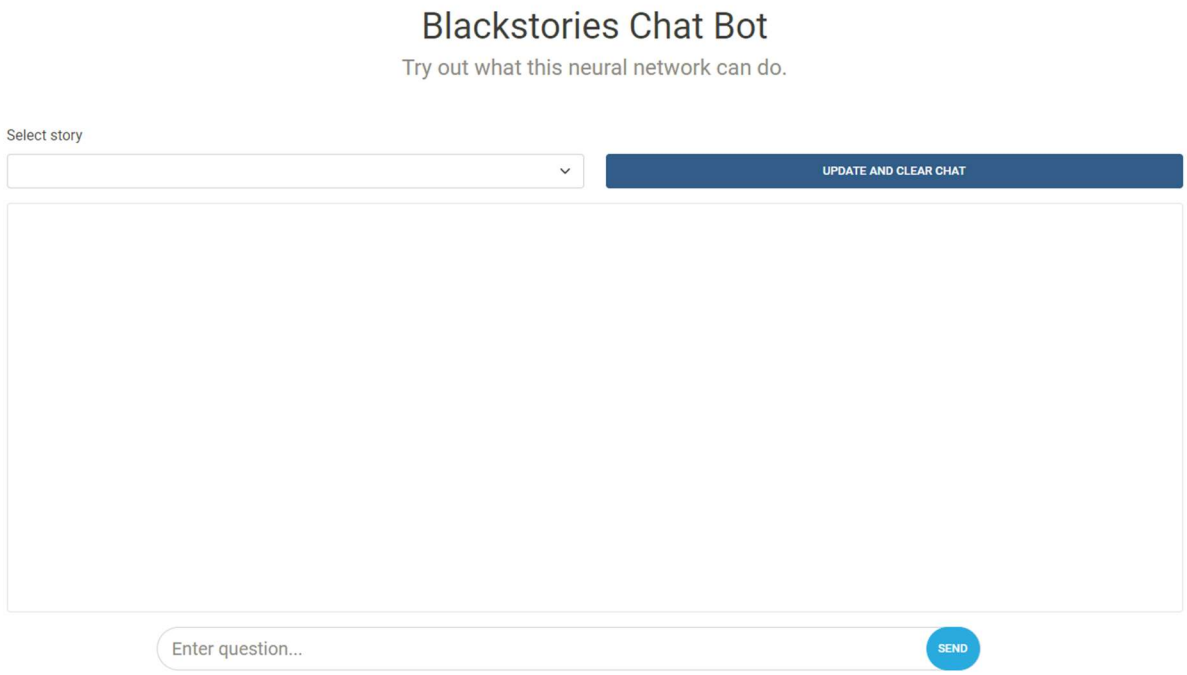



*Abbildung 16: Ansicht der Webapplikation*

Im Dropdown-Menu "Select story" kann aus einer von 9 implementierten Geschichten ausgewählt werden. Die Geschichten sind exemplarische Blackstories [43] und können aufgrund ihres Titels ausgewählt werden. Dann muss man den Chat mit dem Button "update and clear chat" updaten. Folgender Text erscheint:

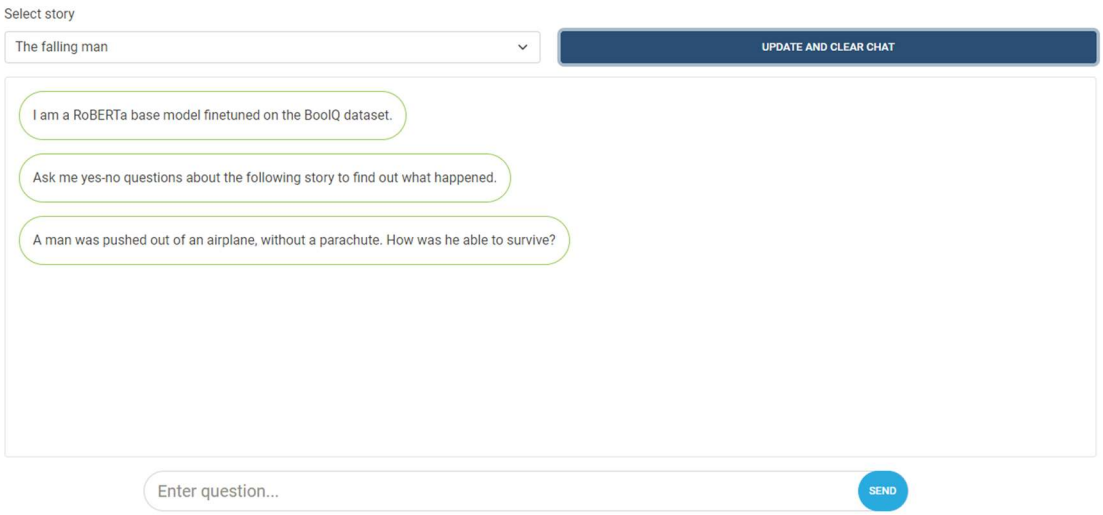



*Abbildung 17: Automatisch generierter Text nach Drücken des Update Buttons*

Die unterste Blase enthält jeweils die Geschichte, die man im Dropdown-Menü ausgewählt hat. Dann kann man im Textfeld unterhalb des Chats Ja-/Nein-Fragen eintippen, die das Modell umgehend beantwortet. Ein möglicher Chatverlauf wäre der Folgende:

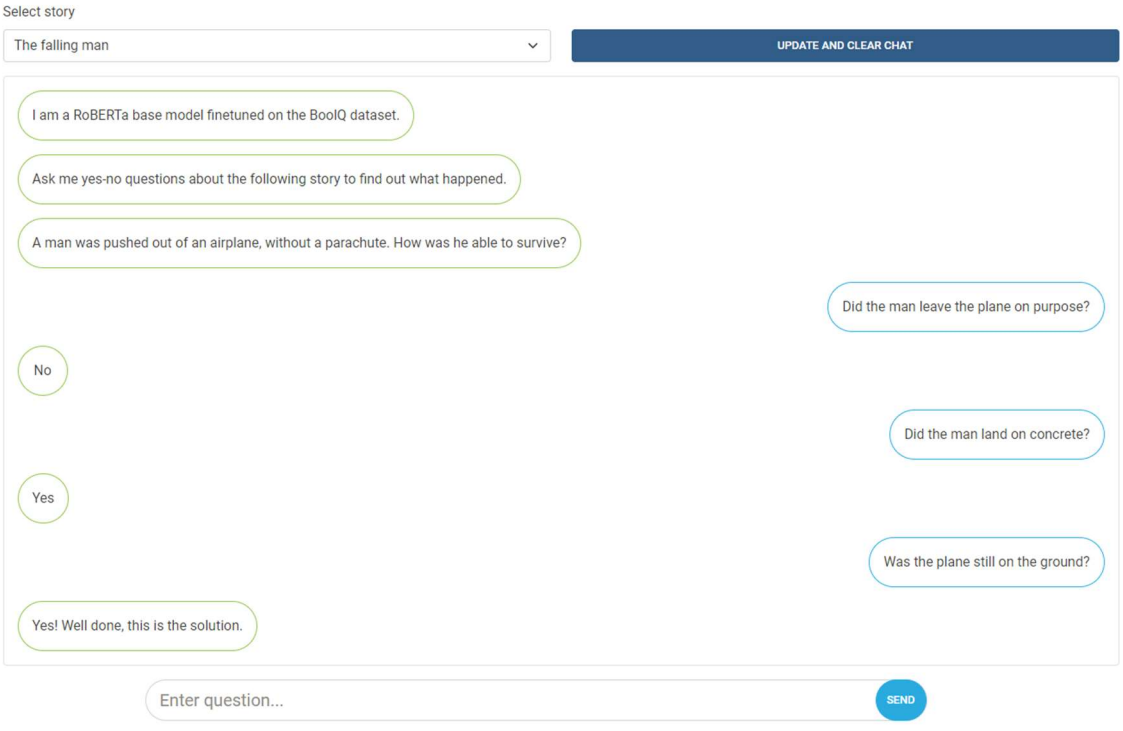



*Abbildung 18: Möglicher Chatverlauf*

Wenn das sekundäre Modell bemerkt, dass der Spieler die Lösung des Rätsels erraten hat, wird nicht mehr nur "Yes" oder "No" geantwortet, sondern dem Spieler wird mitgeteilt, dass das Rätsel gelöst wurde. Um weiterzuspielen, muss man im Dropdown Menü eine neue Geschichte auswählen und den Chat neu updaten.

Zuunterst auf der Seite ist ein Link zum Software-Code der Webseite und den Trainingsskripten zu finden. Die Webseite wurde zusätzlich für kleinere Bildschirme kompatibel gemacht sowie ein Autoscroll eingebaut, damit der untere Rand des Chatverlaufs beim Abschicken einer Frage zu sehen ist. Eine eingegebene Frage kann zudem nicht nur durch das Drücken des "Send" Buttons bestätig werden, sondern auch durch das Drücken der "Enter" Taste.

# 5 Schluss

## 5.1 Ziel erreicht

Ich darf feststellen, dass ich die mir gesetzten Ziele erreicht habe: Der Computer kann als Spielpartner für beliebige Blackstories verwendet werden. Die Interaktion geschieht über eine öffentlich zugängliche Webapplikation und die Antworten des Computers sind so gut und auch so schnell, dass man ihn wirklich als Spielpartner nutzen kann.

## 5.2 Diskussion zur Performance des selbsttrainierten NN

Die Trainingsphase wurde mit dem Training des final verwendeten NN abgeschlossen. Die erreichten Testgenauigkeiten von 73%, 78% und 84% der drei vorgestellten Modelle sind auf den ersten Blick vielleicht nicht sehr eindrucksvoll. Aber verglichen mit den 17 aufgeführten Einträgen im BoolQ Benchmark würden diese drei Modelle immerhin auf den Plätzen 12, 11 und 7 landen. Um die Platzierungen richtig einzuordnen: Die dort aufgeführten Modelle wurden mehrheitlich von grossen Forschungsteams, wie z.B. von Google Research, entwickelt. Abgesehen davon ist der Mensch mit 90% Genauigkeit nur gerade in 5% der Fälle besser, als das hier trainierte Modell Roberta Large. Allerdings hatte ich mir mehr vom Hyperparametertraining des Modells Roberta Base erhofft. Es hat die Performance des ursprünglich trainierten Modells nur knapp übertroffen und dafür dennoch ein Vielfaches der Zeit gebraucht. Diese Ineffektivität des Hyperparametertrainings könnte an zwei Gründen liegen: Entweder wurde im zuerst durchgeführten Training schon eine nahezu optimale Kombination an Hyperparametern ausgewählt und das Modell konnte gar nicht weiter optimiert werden. Oder die Hyperparametersuche würde den vollen Effekt erst bei längerem Training erzielen. Wollte man ansonsten die Resultate dieser Maturaarbeit noch weiterentwickeln, so könnte man andere Basismodelle für das Transferlearning verwenden oder mehr Trainingsdaten durch synthetische Datengenerierung herstellen und verwenden.

## 5.3 Diskussion der Webapplikation

Im Zusammenhang mit der Webapplikation, welche den nach aussen sichtbaren Teil dieser Arbeit darstellt und daher durchaus wichtig ist (auch wenn sie nichts über die Güte der trainierten NN aussagt), musste ich mich mit diversen servertechnischen Aspekten genauer beschäftigen. Dabei habe ich viel gelernt, vor allem das Deployment hatte ich sehr unterschätzt.

Ich war ursprünglich der Meinung, dass ich die App ohne Probleme auf eine Cloudplattform wie Heroku hochladen und dort veröffentlichen könne. Doch beizeiten wurde mir klar, dass ein Server von Heroku, welcher den Anforderungen meiner Webapp gewachsen ist, relativ teuer werden würde: dort hätte die notwendige Hardware-Konfiguration über 250 $ im Monat gekostet! Nach eingehender Recherche fand ich hingegen bei Vultr einen Server für 20 $ im Monat, wobei dieser relativ tiefe Preis allerdings mit einigem Aufwand verbunden ist: so musste ich zum Beispiel den Linux Server selbst aufsetzten und gegen potenzielle Hackerangriffe sichern.

### 5.4 Persönliches Fazit

Durch ausführliche Recherchen und viel Ausprobieren habe ich mich im Verlauf der Arbeit in verschiedenste hochkomplexe Bereiche der NN und der Webentwicklung eingearbeitet. Bei der Entwicklung der Trainingsskripte musste ich vieles austesten und manches hat nicht im ersten Versuch funktioniert. Doch über die Wochen hinweg gewann ich ein solides Verständnis der zugrunde liegenden Konzepte und konnte schlussendlich nicht nur wie geplant den "Spielpartner" designen, sondern auch recht gute Resultate mit meinen NN-Modellen erzielen.
Im Bereich Webentwicklung ging es vergleichsweise schneller voran, da ich in früheren Projekten schon diesbezügliche Erfahrungen gesammelt hatte. Die Webseite mit dem eigens trainierten NN im Hintergrund als finales Produkt ist für mich ein befriedigendes Ergebnis, denn sie funktioniert wirklich und schaut nicht nur schön aus.
Letztlich darf ich feststellen, dass mir das Erarbeiten und auch die Produktion viel Spass bereitet hat und ich Vieles gelernt habe, auf das ich in zukünftigen IT-Projekten sicher gut aufbauen kann.

# Quellenverzeichnis

[1] „Die Geschichte der Künstlichen Intelligenz," [Online]. Available: https://www.bosch.com/de/stories/geschichte-der-kuenstlichen-intelligenz/. [Zugriff am 09 10 2022].

[2] „Machine Learning - Das maschinelle Lernen," [Online]. Available: https://www.arnold-it.com/digitalisierung/machine-learning-maschinelles-lernen. [Zugriff am 07 10 2022].

[3] „Deep Learning: Definition, Beispiele & Frameworks," [Online]. Available: https://datasolut.com/was-ist-deep-learning/. [Zugriff am 15 6 2022].

[4] [Online]. Available: https://www.ki-konkret.de/was-ist-ki.html. [Zugriff am 14 10 2022].

[5] „Black stories," [Online]. Available: https://de.wikipedia.org/wiki/Black_Stories. [Zugriff am 09 10 2022].

[6] [Online]. Available: https://katzlberger.ai/2019/12/13/biologische-und-kuenstliche-neuronen-im-vergleich. [Zugriff am 14 10 2022].

[7] [Online]. Available: https://www.tibco.com/de/reference-center/what-is-a-neural-network. [Zugriff am 14 10 2022].

[8] [Online]. Available: https://www.thingslogic.com/wp-content/uploads/2021/07/Visiosn_Computing_WAeGER_AI.pdf. [Zugriff am 14 10 2022].

[9] „Künstliches Neuron," [Online]. Available: https://de.wikipedia.org/wiki/K%C3%BCnstliches_Neuron. [Zugriff am 25 09 2022].

[10] [Online]. Available: https://de.wikipedia.org/wiki/Tangens_hyperbolicus_und_Kotangens_hyperbolicus. [Zugriff am 14 10 2022].

[11] „Training with Pytorch," [Online]. Available: https://pytorch.org/tutorials/beginner/introyt/trainingyt.html. [Zugriff am 25 09 2022].

[12] „Metrics," [Online]. Available: https://huggingface.co/docs/datasets/how_to_metrics. [Zugriff am 25 09 2022].

[13] „Introduction to Loss Functions," [Online]. Available: https://www.datarobot.com/blog/introduction-to-loss-functions/. [Zugriff am 01 04 2022].

[14] „Overfitting in Machine Learning: What It Is and How to Prevent It," [Online]. Available: https://elitedatascience.com/overfitting-in-machine-learning. [Zugriff am 08 03 2022].

[15] „The 5 Levels of Machine Learning Iteration," [Online]. Available: https://elitedatascience.com/machine-learning-iteration. [Zugriff am 08 03 2022].

[16] „Transfer Learning: Grundlagen und Einsatzgebiete," [Online]. Available: https://datasolut.com/was-ist-transfer-learning/. [Zugriff am 25 09 2022].

[17] „RoBERTa base model," [Online]. Available: https://huggingface.co/roberta-base. [Zugriff am 25 09 2022].

[18] „RoBERTa: A Robustly Optimized BERT Pretraining Approach," [Online]. Available: https://arxiv.org/abs/1907.11692. [Zugriff am 25 09 2022].

[19] „Supermicro NVIDIA Tesla V100-32GB GPU-NVTV100-32," [Online]. Available: https://www.primeline-solutions.com/de/supermicro-nvidia-tesla-v100-32gb-gpu-nvtv100-32/. [Zugriff am 25 09 2022].

[20] „Tokenizer," Huggingface, [Online]. Available: https://huggingface.co/docs/transformers/main_classes/tokenizer. [Zugriff am 30 09 2022].

[21] „Neuronale Netzwerke mit mehreren Klassen: Softmax," [Online]. Available: https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax. [Zugriff am 30 09 2022].

[22] „What Is Argmax in Machine Learning?," [Online]. Available: https://machinelearningmastery.com/argmax-in-machine-learning/. [Zugriff am 30 09 2022].

[23] R. Liaw, „Hyperparameter Search with Transformers and Ray Tune," Raytune, 2 11 2020. [Online]. Available: https://huggingface.co/blog/ray-tune. [Zugriff am 30 09 2022].

[24] K. Li, „How to Choose a Learning Rate Scheduler for Neural Networks," 22 07 2022. [Online]. Available: https://neptune.ai/blog/how-to-choose-a-learning-rate-scheduler. [Zugriff am 30 09 2022].

[25] [Online]. Available: https://neptune.ai/blog/how-to-choose-a-learning-rate-scheduler. [Zugriff am 14 10 2022].

[26] „Difference Between a Batch and an Epoch in a Neural Network," [Online]. Available: https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/. [Zugriff am 10 10 2022].

[27] „TrainingArguments," [Online]. Available: https://huggingface.co/docs/transformers/main_classes/trainer#transformers.TrainingArguments. [Zugriff am 10 10 2022].

[28] „How to Use Random Seeds Effectively," [Online]. Available: https://towardsdatascience.com/how-to-use-random-seeds-effectively-54a4cd855a79. [Zugriff am 10 10 2022].

[29] „Willkommen bei Colab!," Google, [Online]. Available: https://colab.research.google.com/. [Zugriff am 30 09 2022].

[30] „PyTorch," [Online]. Available: https://pytorch.org/. [Zugriff am 01 10 2022].

[31] „Huggingface," [Online]. Available: https://huggingface.co/. [Zugriff am 01 10 2022].

[32] „BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions," [Online]. Available: https://arxiv.org/abs/1905.10044. [Zugriff am 01 10 2022].

[33] „Boolq Huggingface," [Online]. Available: https://huggingface.co/datasets/boolq. [Zugriff am 01 10 2022].

[34] „wandb," [Online]. Available: https://wandb.ai/home. [Zugriff am 01 10 2022].

[35] „Summary of the models," Huggingface, [Online]. Available: https://huggingface.co/docs/transformers/model_summary. [Zugriff am 01 10 2022].

[36] „bert-base-uncased,“ [Online]. Available: https://huggingface.co/bert-base-uncased. [Zugriff am 01 10 2022].

[37] „Dash Python User Guide,“ [Online]. Available: https://dash.plotly.com/. [Zugriff am 01 10 2022].

[38] „Dash Bootstrap Components,“ [Online]. Available: https://dash-bootstrap-components.opensource.faculty.ai/. [Zugriff am 01 10 2022].

[39] „CSS Introduction,“ [Online]. Available: https://www.w3schools.com/css/css_intro.asp. [Zugriff am 01 10 2022].

[40] „The Infrastructure Cloud™,“ [Online]. Available: https://www.vultr.com/. [Zugriff am 01 10 2022].

[41] „Was ist ein SSL Zertifikat?,“ [Online]. Available: https://www.globalsign.com/de-de/ssl-information-center/was-ist-ein-ssl-zertifikat. [Zugriff am 01 10 2022].

[42] „SuperGLUE benchmark leaderboard,“ [Online]. Available: https://super.gluebenchmark.com/leaderboard. [Zugriff am 09 10 2022].

[43] „Black Stories,“ [Online]. Available: https://www.cram.com/flashcards/black-stories-3539866. [Zugriff am 09 10 2022].

[44] „Instantiating a big model,“ Huggingface, [Online]. Available: https://huggingface.co/docs/transformers/big_models. [Zugriff am 30 09 2022].

[45] „ST-MoE: Designing Stable and Transferable Sparse Expert Models,“ [Online]. Available: https://arxiv.org/abs/2202.08906v2. [Zugriff am 14 10 2022].

# Abbildungsverzeichnis



Hinweis: Alle Abbildungen ohne dortiger Quellenangabe wurden selbst erstellt (z.T. unter Zuhilfenahme der Software Wandb)

## Glossar mit Kurzerklärungen

**API**: Das "Application Programming Interface", kurz API, ist eine Schnittstelle, die es erlaubt Daten zu transferieren.

**Benchmark**: Eine "Benchmark" ist ein Vergleichsmassstab, welcher hier dafür verwendet wird, die Performance Neuronaler Netzte vergleichbar zu machen.

**DNS-Server**: Der "Domain Name System"-Server, kurz DNS-Server, ordnet jeder IP-Adresse eine URL-Adresse zu und umgekehrt.

**Domänenspezifisch**: Auf eine spezifische Anwendung spezialisiert.

**Effektive Lernrate**: Die Lernrate kann während eines Trainings variieren. Die effektive Lernrate stellt die Lernrate des Optimieres zum Zeitpunkt jedes einzelnen Beispiels dar.

**Exemplarische Blackstory**: (Wobei die auf der Webseite verwendeten Blackstories nur in der englischen Version mit englischen Fragen funktionieren.) Geschichte: "Ein Mann wurde ohne Fallschirm aus einem Flugzeug gestossen. Wie konnte er überleben?" Auflösung: "Das Flugzeug war noch gar nicht in der Luft. Es stand immer noch auf dem Flugplatz."

**Hosting**: Das Hosting ist ein Dienstleistungsangebot für das Bereitstellen, Unterstützen und Veröffentlichen von Webapplikationen.

**IP-Adresse**: Die IP-Adresse ist das Identifikationsmerkmal von jedem Computer im Internet.

**Metrik**: Eine Metrik ist ein quantitativer Messwert, der während einem Messvorgang gemessen wird.

**NLP**: Natural Language Processing (siehe Kapitel 1.1)

**NN**: Neuronales Netz (siehe Kapitel 2.1.1)

**Pseudozufällig**: Eine Pseudozufälligkeit ist keine richtige Zufälligkeit. Computer sind deterministisch und können gar keine richtigen Zufallszahlen generieren. Sie generieren darum mit mathematischen Funktionen pseudozufällige Zahlen, die für die meisten Anwendungen zufällig genug sind.

**SSL**: SSL, auch TLS genannt, ist ein Protokoll für die Verschlüsselung von Webdaten. Es erlaubt die Nutzung von sicheren Https-Verbindungen anstatt unsicheren Http-Verbindungen.

**Ubuntu**: Ubuntu ist ein Linux Betriebssystem, welches oft auf Servern verwendet wird.

**UI**: User Interface (siehe Kapitel 2.2.1)

**URL-Adresse**: Ein "Uniform Resource Locator", kurz URL, ist eine Adresse für jegliche Ressourcen auf dem Internet.

Hiermit bestätige ich, dass ich meine Maturitätsarbeit selbstständig und nur unter Zuhilfenahme der in den Verzeichnissen oder in den Anmerkungen genannten Quellen angefertigt habe. Die Mitwirkung von anderen Personen hat sich auf Beratung und Korrekturlesen beschränkt. Alle verwendeten Unterlagen und Gewährspersonen sind vollständig aufgeführt.

Ort: Datum: Unterschrift: